APPLICATION NOTE

AP-276

June 1986

Implementing a CMOS Bus Arbiter/Controller in the 5C060 EPLD

DANIEL E. SMITH
APPLICATIONS ENGINEERING
INTEL CORPORATION

Order Number: 292012-001

INTRODUCTION

This application note shows how to implement a CMOS Bus Arbiter/Controller in an Intel 5C060 EPLD (Erasable Programmable Logic Device). The note includes a brief overview of a similar circuit implemented with typical PLA devices, a more detailed discussion of the 5C060 implementation, and a summary.

The bus priority resolution and arbitration scheme selected for the circuit is that used by the industry-standard MULTIBUS I interface. Operation and timing for the MULTIBUS I interface is well understood by most engineers and is described in readily available Intel publications. Thus, a description of the MULTIBUS I interface is not included here. The bus arbiter/controller functions shown here support both serial and parallel priority resolution between bus masters. Timing is equivalent to MULTIBUS I specifications. Electrical specifications for both the PLA and EPLD approaches vary from MULTIBUS I standards. Neither of the two circuits discussed here provide the full current sink capability for all MULTIBUS I signals. Because the EPLD implementation is designed for CMOS systems, however, this requirement is not relevant for the 5C060 implementation.

PLA APPROACH

The functional equivalent of a MULTIBUS I arbiter/controller can be implemented in two 20-pin PLA-type devices as shown in Figures 1 and 2. (Figure 1 shows the logic for the arbiter device. Figure 2 shows the logic for the controller and the connections to the arbiter.) Figure 3 shows the arbiter list file as an example of PLA-type files. Two different 20-pin PLA devices are required to implement the arbiter and controller functions, a 16R4-type device and a 16L8-type device.

Implementation of logic devices in PLA-type devices, such as those shown here, has proven to be quite beneficial. Development time and cost is much less than for custom silicon device designs. The two PLA-type devices take up less board space than a discrete TTL implementation of the same functions. In addition, the two raw devices can also be used for different functions in other products, thereby reducing inventory costs. As a result of these factors (and others), use of PLA-type devices has grown substantially in recent years.

With the increased density and flexibility of EPLD devices over typical PLA-type devices, even greater space, inventory, and cost savings can be obtained by using EPLDs. The following section shows an implementation of the same arbiter/controller functions in a single 24-pin 5C060 EPLD device.

5C060 IMPLEMENTATION

The equivalent functions for both the MULTIBUS I arbiter and controller fit inside a single 5C060 EPLD device. The 5C060 device is available in a 24-pin 0.3" DIP package. Figures 4 and 5 show logic diagrams for the arbiter and controller functions. When compared with the PLA implementation, some differences in the design are immediately apparent. These differences result from the characteristics of the EPLD macrocell or from corrections to the circuit used in Figures 1 and 2.

The major change resulting from the EPLD macrocell structure concerns the EPLD output buffers. Since output buffers from macrocells are non-inverting (PLA-type devices typically contain inverting buffers), signals enter the buffers in the same logic orientation from which they are to appear at the output. The logic for the EPLD (shown in Figures 4 and 5) incorporates this change.

Some errors in the PLA-type implementation have also been corrected in the EPLD design. These changes are as follows:

- The M/$\overline{IO}$ input to the MRDC/ and MWTC/ gates is inverted. M/$\overline{IO}$ distinguishes between memory and I/O cycles. The PLA-type implementation does not use this signal properly; the PLA-type controller generates read or write commands to both memory and I/O at the same time, which can result in contention between memory and I/O during bus transfers.
- BPRO/ is gated by BPRN/ in the EPLD design. When using serial priority resolution, this allows the highest priority arbiter to prevent all other masters from controlling the bus. (In the PLA design, BPRO/ is enabled/disabled only by a local request. Higher priority arbiters cannot disable all other arbiters. This can result in contention between bus masters. By gating BPRO/ with BPRN/ in the EPLD design, this source of bus contention is prevented.)

Figure 6 shows the list file for the arbiter/controller device. Figure 7 shows the report file produced by the iPLDS software. This file contains a pinout diagram of the final programmed device and provides a resource usage map for the device.

Most of the input and output signals are self-explanatory to those familiar with Intel processors and the MULTIBUS I interface. The XREQ input is the bus transfer request signal from the address decode logic. The BUSY/ and CBRQ/ outputs are bi-directional, simulated open-collector outputs. These outputs use the iPLDS 5C060 (Combinational-Output I/O-Feedback) primitive in the list file. The BUSY/ signal serves to illustrate this use of EPLD outputs.

A pull-up resistor is used externally (i.e., on the backplane) to hold BUSY/ high when no arbiter is in control of the bus. When the arbiter is granted control of the bus, AEN is clocked high, which enables the output of the BUSY/ driver. Since the input to the BUSY/ driver is low during normal operation (RESET/ inverted), the enabled driver pulls BUSY/ low to signal other arbiters that the bus is in use. When the arbiter is finished using the bus, AEN goes low to disable the BUSY/ driver (three-state output). The pull-up resistor pulls BUSY/ high to signal other arbiters that the bus is free for use if needed.

Note that BUSY/ is also routed into the bus grant logic as input BSI. BSI prevents the arbiter from taking control of the bus (and driving BUSY/ low) when some other arbiter already has control of the bus. Thus only one arbiter may pull BUSY/ low at any one time.

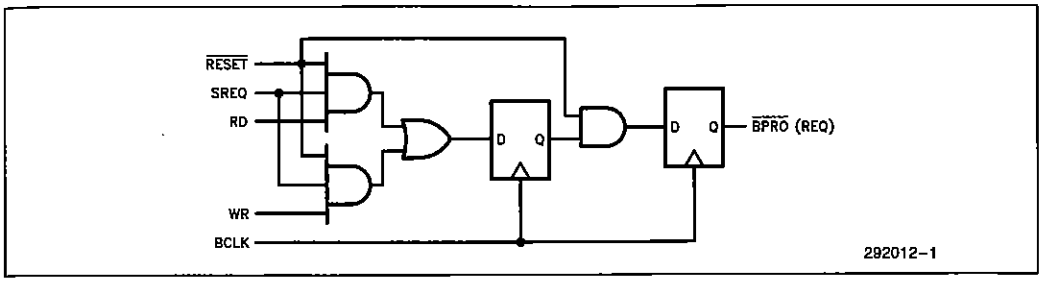
The one difference between standard MULTIBUS I logic levels and the EPLD implementation described here relates to the BCLK/ signal. MULTIBUS I bus arbitration uses the negative-going edge of BCLK/ to synchronize events. All 5C060 flip-flops, however, clock on the positive-going edge of BCLK/. If all bus masters in the system use the same arbiter implementation, this poses no problem. Otherwise, an external inverter is required for the BCLK/ input.

COMPARISON/SUMMARY

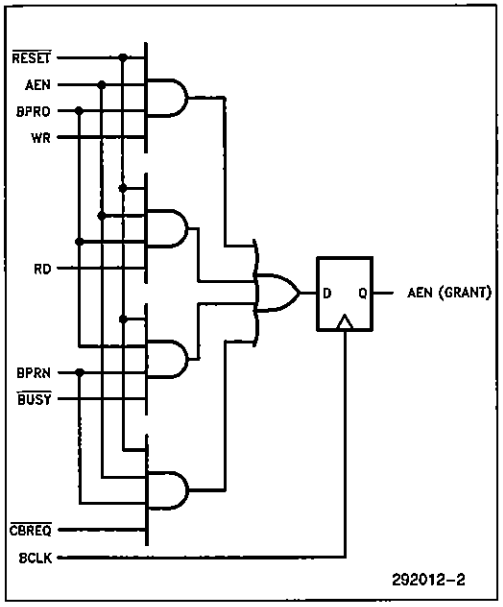
Both the PLA and EPLD implementations of the bus arbiter/controller result in a lower device count than a discrete logic circuit. Lower device count means less p.c. board space, fewer assembly steps, and fewer device interconnects. Both PLA and EPLD implementations are quicker and less expensive to develop than a custom gate array or dedicated silicon device.

In contrast to the PLA approach, however, the EPLD implementation requires only a single device, while the PLA approach requires two different devices. Thus the EPLD approach results in twice the cost savings (inventory and assembly) and half the programming activity to produce the device. Fewer device interconnects also means greater reliability. In addition, programmed EPLD devices can be erased and reprogrammed for a different application if needed, a feature not available with PLAs.

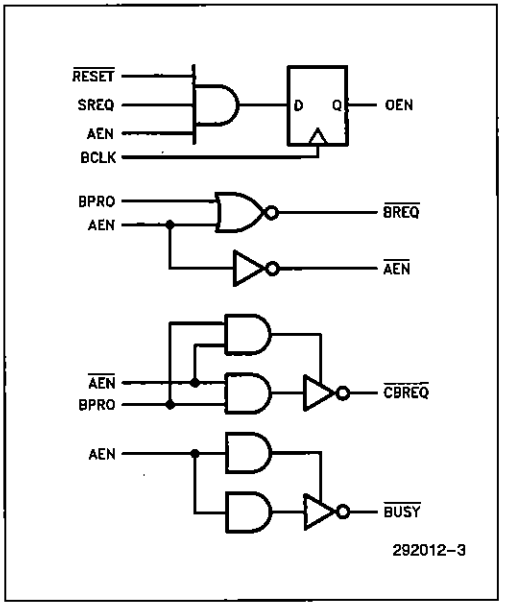
Overall, the greater flexibility, and the incremental design, manufacturing, and cost advantages of EPLD devices make them ideal for many applications where PLA devices would otherwise be used.



A) Request Synchronizer



B) Grant/Access Logic



C) Bus Transfer Control

Figure 1. PLA Approach to a Bus Arbiter

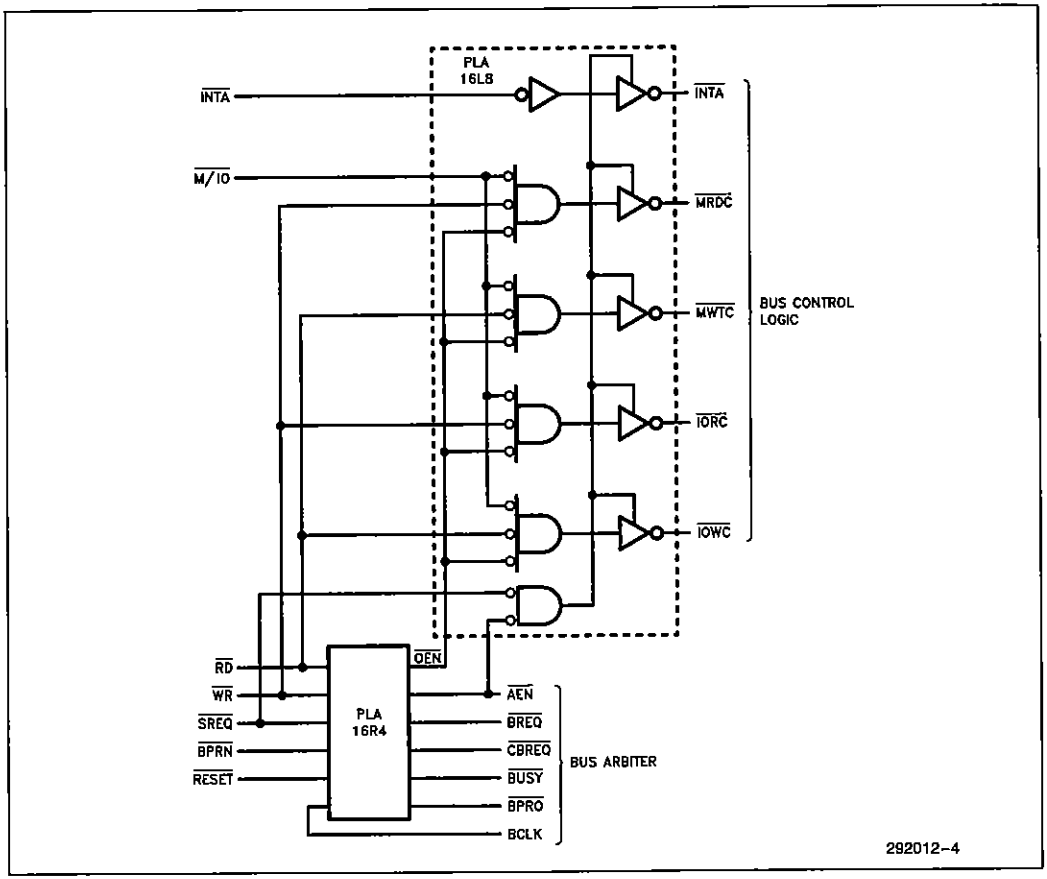
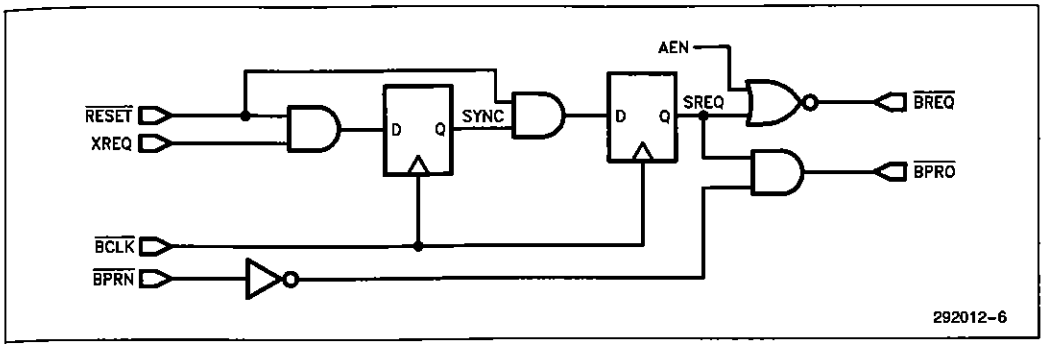


Figure 2. Bus Controller with Arbiter Connected

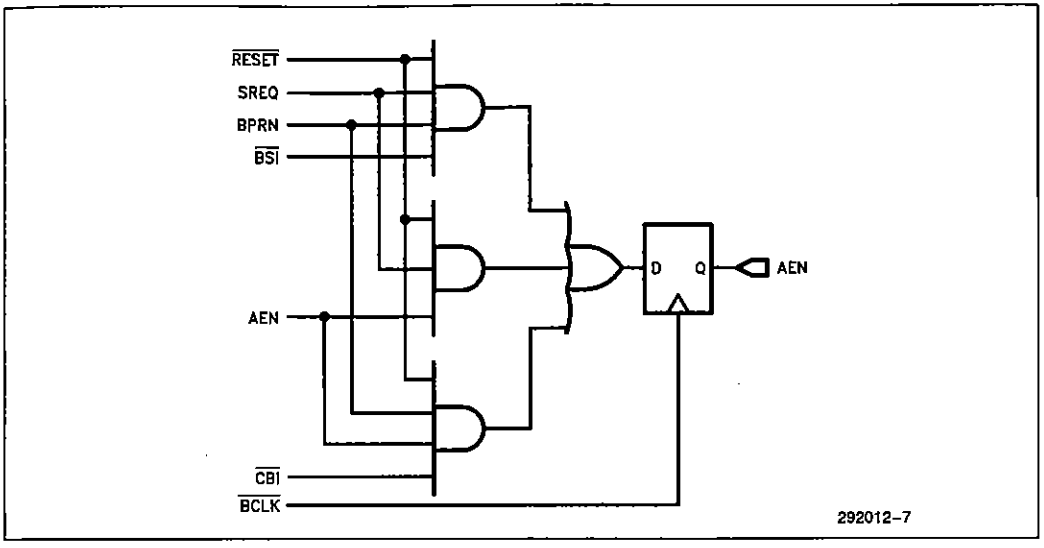
```
PLA16R4                                              PLA DESIGN FILE
ARB001                                            D. E. ENGR.  1/1/85
MULTIBUS I ARBITER
SOME SYSTEM COMPANY
BCLK  /WR    /RD   /SREQ /RESET /BPRN  NC   NC    NC  GND
/E    /CBREQ /BUSY /SYNC /BPRO  /AEN   /OEN /BREQ NC  VCC

SYNC  := /RESET*SREQ*WR          +
         /RESET*SREQ*RD

BPRO  := /RESET*SYNC

AEN   := /RESET* AEN*BPRO*WR     +
         /RESET* AEN*BPRO*RD     +
         /RESET*BPRO*BPRN*/BUSY  +
         /RESET* AEN*BPRN*/CBREQ

OEN   := /RESET*SREQ*AEN

IF(BPRO*/AEN) CBREQ = BPRO*/AEN

IF(AEN) BUSY = AEN

BREQ   = BPRO +
         AEN
```

292012-5

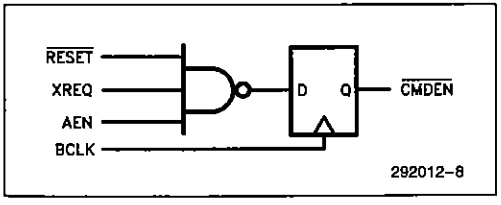
Figure 3. List File for PLA Arbiter



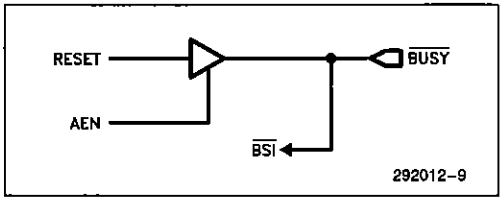
A) Request



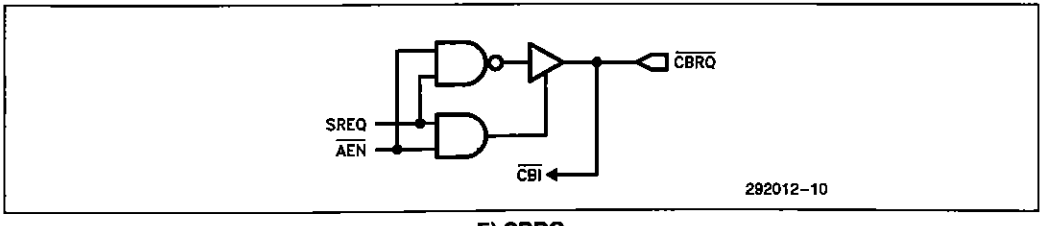
B) Grant



C) Command Enable



D) Busy



E) CBRQ

Figure 4. Logic Diagram of Bus Arbiter Functions

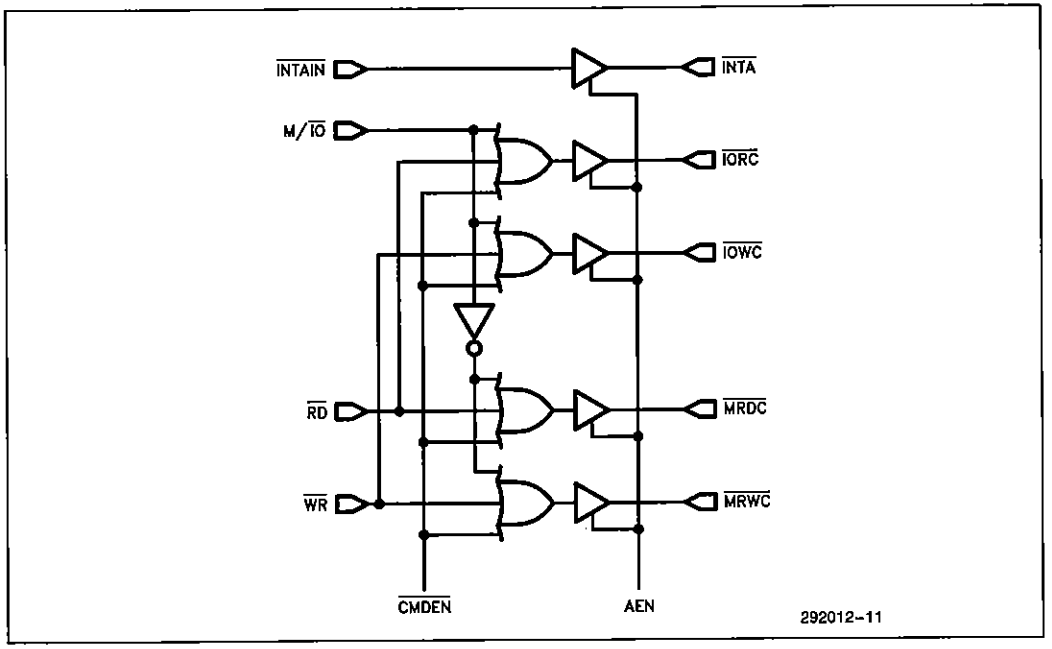


Figure 5. Logic Diagram of Bus Controller Functions

```
DANIEL E. SMITH
INTEL CORPORATION
MARCH 27, 1986
VERSION 1.1
REV. A
5C060
CMOS BUS ARBITER/CONTROLLER


PART:      5C060
INPUTS:    BCLK, XREQ, RESET, BPRN, MIO,  RD,   WR,   INTAIN
OUTPUTS:   BPRO, AEN,  BREQ,  CBRQ, BUSY, INTA, MRDC, MWTC, IORC, IOWC


NETWORK:

BCLK      = INP (BCLK)                    %BUS CLOCK INPUT%
INTAIN    = INP (INTAIN)                  %INT. ACK. INPUT%
XREQ      = INP (XREQ)                    %SYSTEM REQUEST INPUT%
RESET     = INP (RESET)                   %RESET INPUT%
BPRN      = INP (BPRN)                    %BUS PRIORITY INPUT%
MIO       = INP (MIO)                     %MEMORY/IO INPUT%
RD        = INP (RD)                      %READ INPUT%
WR        = INP (WR)                      %WRITE INPUT%
BPRO      = CONF (BPROc,VCC)              %BUS PRIORITY OUTPUT%
AEN,AEN   = RORF (AENd,BCLK,GND,GND,VCC) %ADDRESS ENABLE (GRANT)%
BREQ      = CONF (BREQc,VCC)              %BUS REQUEST%
CBRQ,CBI  = COIF (CBRQc1,CBRQc2)          %CBRQ/ -- SIMULATED O.C.%
BUSY,BSI  = COIF (BUSYc,AEN)              %BUSY/ -- SIMULATED O.C.%
INTA      = CONF (INTAIN,AEN)             %INT. ACK. OUTPUT%
MRDC      = CONF (MRDCc,AEN)              %MEMORY READ COMMAND%
MWTC      = CONF (MWTCc,AEN)              %MEMORY WRITE COMMAND%
IORC      = CONF (IORCc,AEN)              %I/O READ COMMAND%
IOWC      = CONF (IOWCc,AEN)              %I/O WRITE COMMAND%
SREQ      = NORF (SREQd,BCLK,GND,GND)     %VALID BUS REQUEST%
SYNC      = NORF (SYNCd,BCLK,GND,GND)     %SYNCHRONIZED REQUEST%
CMDEN     = NORF (CMDENd,BCLK,GND,GND)    %COMMAND ENABLE%
```

292012-12

```
EQUATIONS:

BPROc     = (SREQ * /BPRN);
AENd      = RESET * SREQ * /BPRN * BSI +
            RESET * SREQ * AEN +
            RESET * /BPRN * AEN * CBI;
BREQc     = /(SREQ + AEN);
BUSYc     = /RESET;
CBRQc1    = /(SREQ * /AEN);
CBRQc2    = SREQ * /AEN;
MRDCc     = /MIO + RD + CMDEN;
MWTCc     = /MIO + WR + CMDEN;
IORCc     = MIO + RD + CMDEN;
IOWCc     = MIO + WR + CMDEN;
SREQd     = RESET * SYNC;
SYNCd     = RESET * XREQ;
CMDENd    = /(RESET * XREQ * AEN);


END$
```

292012-13

Figure 6. IPLDS Network List File

```
Logic Optimizing Compiler Utilization Report

 ***** Design implemented successfully

DANIEL E. SMITH
INTEL CORPORATION
MARCH 27, 1986
VERSION 1.1
REV. A
5C060
CMOS BUS ARBITER/CONTROLLER


                5C060
             - - - - -
      BCLK -: 1    24:- Vcc
       MIO -: 2    23:- XREQ
  RESERVED -: 3    22:- INTA
  RESERVED -: 4    21:- IOWC
  RESERVED -: 5    20:- IORC
       AEN -: 6    19:- MWTC
      BPRO -: 7    18:- MRDC
    INTAIN -: 8    17:- BUSY
        WR -: 9    16:- CBRQ
        RD -:10    15:- BREQ
      BPRN -:11    14:- RESET
       GND -:12    13:- GND
             - - - - -


**INPUTS**

                                                  :           Feeds:
      Name  Pin  Resource   MCell #    PTerms     :  MCells     OE  Clear  Clock

      BCLK   1      INP        -         -              -       -     -    CLK1

       MIO   2      INP        -         -              2       -     -     -
                                                        3
                                                        4
                                                        5

    INTAIN   8      INP       14       0/ 8             1       -     -     -

        WR   9      INP       15       0/ 8             2       -     -     -
                                                        4

        RD  10      INP       16       0/ 8             3       -     -     -
                                                        5

      BPRN  11      INP        -         -             12       -     -     -
                                                       13

     RESET  14      INP        -         -              6       -     -     -
                                                        9
                                                       10
                                                       11
                                                       12

      XREQ  23      INP        -         -              9       -     -     -
                                                       10
```

292012-14

Figure 7. IPLDS Report File

OUTPUTS

Name	Pin	Resource	MCell #	PTerms	:	Feeds: MCells	OE	Clear	Clock
AEN	6	NORF	12	3/ 8		7 8 9 12	-7 1 2 3 4 6 6	-	-
BPRO	7	CONF	13	1/ 8		-	-	-	-
BREQ	15	CONF	8	1/ 8		-	-	-	-
CBRQ	16	COIF	7	1/ 8		12	-	-	-
BUSY	17	COIF	6	1/ 8		12	-	-	-
MRDC	18	CONF	5	1/ 8		-	-	-	-
MWTC	19	CONF	4	1/ 8		-	-	-	-
IORC	20	CONF	3	1/ 8		-	-	-	-
IOWC	21	CONF	2	1/ 8		-	-	-	-
INTA	22	CONF	1	1/ 8		-	-	-	-

BURIED REGISTERS

Name	Pin	Resource	MCell #	PTerms	:	Feeds: MCells	OE	Clear	Clock
	3	NORF	9	1/ 8		2 3 4 5	-	-	-
	4	NORF	10	1/ 8		11	-	-	-
	5	NORF	11	1/ 8		7 8 12 13	7	-	-

UNUSED RESOURCES

Name	Pin	Resource	MCell	PTerms
-	13	-	-	-

PART UTILIZATION

95% Pins
100% MacroCells
11% Pterms

292012-15

Figure 7. IPLDS Report File (Continued)